
INVENTORY
SQL PROJECT

OBJECTIVE

THE OBJECTIVE OF THIS PROJECT IS TO PROVIDE A COMPREHENSIVE ANALYSIS OF PRODUCTION AND STORE PERFORMANCE BY LEVERAGING KEY METRICS AND TRENDS. THIS INCLUDES IDENTIFYING TOP-PERFORMING PRODUCTS BASED ON TOTAL SALES AND PROFIT, EVALUATING SALES PERFORMANCE AND PROFIT MARGINS ACROSS DIFFERENT STORES, AND EXAMINING MONTHLY SALES TRENDS USING A ROLLING 3-MONTH AVERAGE TO UNCOVER PERIODS OF SIGNIFICANT GROWTH OR DECLINE. ADDITIONALLY, THE PROJECT AIMS TO CALCULATE THE CUMULATIVE DISTRIBUTION OF PROFIT MARGINS FOR PRODUCT CATEGORIES TO IDENTIFY AREAS OF PROFITABILITY AND ANALYZE STORE EFFICIENCY BY CALCULATING THE INVENTORY TURNOVER RATIO, PROVIDING ACTIONABLE INSIGHTS TO OPTIMIZE OPERATIONS AND IMPROVE OVERALL BUSINESS PERFORMANCE.

IDENTIFY TOP-PERFORMING PRODUCTS BASED ON TOTAL SALES AND PROFIT

```sql
SELECT
    p.Product_ID,
    p.Product_Name,
    p.Product_Category,
    SUM(s.Units) AS Total_Units_Sold,
    SUM(s.Units * CAST(REPLACE(TRIM(p.Product_Price), '$', '') AS NUMERIC)) AS Total_Revenue,
    SUM(s.Units * (CAST(REPLACE(TRIM(p.Product_Price), '$', '') AS NUMERIC) -
                   CAST(REPLACE(TRIM(p.Product_Cost), '$', '') AS NUMERIC))) AS Total_Profit
FROM
    Sales s
JOIN
    Products p ON s.Product_ID = p.Product_ID
GROUP BY
    p.Product_ID, p.Product_Name, p.Product_Category
ORDER BY
    Total_Revenue DESC, Total_Profit DESC
LIMIT 10;
```

product_id [PK] integer	product_name character varying (100)	product_category character varying (50)	total_units_sold bigint	total_revenue numeric	total_profit numeric
18	Lego Bricks	Toys	59737	2388882.63	298685.00
6	Colorbuds	Electronics	104368	1564476.32	834944.00
19	Magic Sand	Art & Crafts	60598	968962.02	121196.00
1	Action Figure	Toys	57958	926748.42	347748.00
30	Rubik's Cube	Games	45672	912983.28	91344.00
8	Deck Of Cards	Games	84034	587397.66	252102.00
31	Splash Balls	Sports & Outdoors	60248	541629.52	60248.00
24	Nerf Gun	Sports & Outdoors	26543	530594.57	132715.00
2	Animal Figures	Toys	39089	507766.11	117267.00
7	Dart Gun	Sports & Outdoors	31588	505092.12	126352.00

ANALYSE SALES PERFORMANCE FOR EACH STORE, INCLUDING TOTAL REVENUE AND PROFIT MARGIN

```sql
SELECT
    st.Store_ID,
    st.Store_Name,
    st.Store_City,
    SUM(s.Units * CAST(REGEXP_REPLACE(p.Product_Price, '[^\d.]+', '', 'g') AS NUMERIC)) AS Total_Revenue,
    SUM(s.Units * (CAST(REGEXP_REPLACE(p.Product_Price, '[^\d.]+', '', 'g') AS NUMERIC) -
                   CAST(REGEXP_REPLACE(p.Product_Cost, '[^\d.]+', '', 'g') AS NUMERIC))) AS Total_Profit,
    ROUND(
        CASE
            WHEN SUM(s.Units * CAST(REGEXP_REPLACE(p.Product_Price, '[^\d.]+', '', 'g') AS NUMERIC)) > 0 THEN
                (SUM(s.Units * (CAST(REGEXP_REPLACE(p.Product_Price, '[^\d.]+', '', 'g') AS NUMERIC) -
                                CAST(REGEXP_REPLACE(p.Product_Cost, '[^\d.]+', '', 'g') AS NUMERIC))) /
                 SUM(s.Units * CAST(REGEXP_REPLACE(p.Product_Price, '[^\d.]+', '', 'g') AS NUMERIC))) * 100
            ELSE 0
        END, 2
    ) AS Profit_Margin
FROM
    Sales s
JOIN
    Products p ON s.Product_ID = p.Product_ID
JOIN
    Stores st ON s.Store_ID = st.Store_ID
GROUP BY
    st.Store_ID, st.Store_Name, st.Store_City
ORDER BY
    Total_Revenue DESC;
```

store_id [PK] integer	store_name character varying (100)	store_city character varying (50)	total_revenue numeric	total_profit numeric	profit_margin numeric
31	Toys Ciudad de Mexico 2	Cuidad de Mexico	554553.43	169856.00	30.63
30	Toys Guadalajara 3	Guadalajara	449354.91	121571.00	27.05
9	Toys Ciudad de Mexico 1	Cuidad de Mexico	433556.21	111296.00	25.67
17	Toys Toluca 1	Toluca	411157.32	104612.00	25.44
7	Toys Monterrey 2	Monterrey	372998.82	106783.00	28.63
46	Toys Guadalajara 4	Guadalajara	348466.64	102178.00	29.32
42	Toys Hermosillo 3	Hermosillo	344846.64	98825.00	28.66
39	Toys Xalapa 2	Xalapa	344307.04	88637.00	25.74
37	Toys Ciudad de Mexico 3	Cuidad de Mexico	337424.66	94021.00	27.86
4	Toys Saltillo 1	Saltillo	330408.90	94252.00	28.53
47	Toys Monterrey 4	Monterrey	325073.50	79339.00	24.41
45	Toys Ciudad de Mexico 4	Cuidad de Mexico	323957.71	90385.00	27.90
41	Toys Hermosillo 2	Hermosillo	323427.02	87995.00	27.21
14	Toys Guanajuato 1	Guanajuato	313916.60	88002.00	28.03
10	Toys Campeche 1	Campeche	311786.44	88248.00	28.30
25	Toys Ciudad Victoria 1	Ciudad Victoria	294803.99	83088.00	28.18
6	Toys Mexicali 1	Mexicali	294019.42	97206.00	33.06
13	Toys Mexicali 2	Mexicali	292156.43	77842.00	26.64
33	Toys Monterrey 3	Monterrey	285814.24	86622.00	30.31
28	Toys Puebla 2	Puebla	282616.87	75082.00	26.57
22	Toys Guanajuato 2	Guanajuato	278926.67	79550.00	28.52
2	Toys Monterrey 1	Monterrey	277959.14	73985.00	26.62
21	Toys Santiago 1	Santiago	277598.14	72922.00	26.27

EXAMINE MONTHLY SALES TRENDS, CONSIDERING THE ROLLING 3-MONTH AVERAGE AND IDENTIFYING MONTHS WITH SIGNIFICANT GROWTH OR DECLINE

```sql
WITH Monthly_Sales AS (
    SELECT
        TO_CHAR(Date, 'YYYY-MM') AS Month,
        SUM(Units) AS Monthly_Total_Units
    FROM Sales
    GROUP BY TO_CHAR(Date, 'YYYY-MM')
    ORDER BY TO_CHAR(Date, 'YYYY-MM')
),
Rolling_Avg AS (
    SELECT
        Month,
        Monthly_Total_Units,
        ROUND(
            AVG(Monthly_Total_Units) OVER (ORDER BY Month ROWS BETWEEN 2 PRECEDING AND CURRENT ROW),
            2
        ) AS Rolling_3_Month_Avg
    FROM Monthly_Sales
),
```

```sql
Growth_Decline AS (
    SELECT
        Month,
        Monthly_Total_Units,
        Rolling_3_Month_Avg,
        LAG(Monthly_Total_Units) OVER (ORDER BY Month) AS Previous_Month_Units,
        CASE
            WHEN LAG(Monthly_Total_Units) OVER (ORDER BY Month) IS NOT NULL THEN
                ROUND(
                    ((Monthly_Total_Units - LAG(Monthly_Total_Units) OVER (ORDER BY Month))::NUMERIC
                    / LAG(Monthly_Total_Units) OVER (ORDER BY Month)) * 100, 2
                )
            ELSE NULL
        END AS Percentage_Change
    FROM Rolling_Avg
)
SELECT
    Month,
    Monthly_Total_Units,
    Rolling_3_Month_Avg,
    Percentage_Change,
    CASE
        WHEN Percentage_Change > 20 THEN 'Significant Growth'
        WHEN Percentage_Change < -20 THEN 'Significant Decline'
        ELSE 'Stable'
    END AS Growth_Trend
FROM Growth_Decline
ORDER BY Month;
```

month text	monthly_total_units bigint	rolling_3_month_avg numeric	percentage_change numeric	growth_trend text
2022-01	38009	38009.00	[null]	Stable
2022-02	36935	37472.00	-2.83	Stable
2022-03	39981	38308.33	8.25	Stable
2022-04	47102	41339.33	17.81	Stable
2022-05	46910	44664.33	-0.41	Stable
2022-06	47178	47063.33	0.57	Stable
2022-07	46177	46755.00	-2.12	Stable
2022-08	39927	44427.33	-13.53	Stable
2022-09	42196	42766.67	5.68	Stable
2022-10	47861	43328.00	13.43	Stable
2022-11	51185	47080.67	6.95	Stable
2022-12	66031	55025.67	29.00	Significant Growth
2023-01	56478	57898.00	-14.47	Stable
2023-02	54581	59030.00	-3.36	Stable
2023-03	69336	60131.67	27.03	Significant Growth
2023-04	65593	63170.00	-5.40	Stable
2023-05	63851	66260.00	-2.66	Stable
2023-06	64275	64573.00	0.66	Stable
2023-07	63614	63913.33	-1.03	Stable
2023-08	51193	59694.00	-19.53	Stable
2023-09	52152	55653.00	1.87	Stable

CALCULATE THE CUMULATIVE DISTRIBUTION OF PROFIT MARGIN FOR EACH PRODUCT CATEGORY, CONSIDER WHERE PRODUCTS ARE HAVING PROFIT

```sql
WITH ProfitData AS (
    SELECT
        s.Sale_ID,
        s.Product_ID,
        p.Product_Category,
        (CAST(REGEXP_REPLACE(p.Product_Price, '[^0-9.]', '', 'g') AS NUMERIC) -
         CAST(REGEXP_REPLACE(p.Product_Cost, '[^0-9.]', '', 'g') AS NUMERIC)) * s.Units AS Profit
    FROM
        Sales s
    JOIN
        Products p ON s.Product_ID = p.Product_ID
    WHERE
        CAST(REGEXP_REPLACE(p.Product_Price, '[^0-9.]', '', 'g') AS NUMERIC) >
        CAST(REGEXP_REPLACE(p.Product_Cost, '[^0-9.]', '', 'g') AS NUMERIC)
),
CategoryProfit AS (
    SELECT
        Product_Category,
        SUM(Profit) AS Total_Profit
    FROM
        ProfitData
    GROUP BY
        Product_Category
),
```

```sql
CumulativeDistribution AS (
    SELECT
        Product_Category,
        Total_Profit,
        SUM(Total_Profit) OVER (ORDER BY Total_Profit) AS Cumulative_Profit,
        SUM(Total_Profit) OVER () AS Overall_Profit,
        SUM(Total_Profit) OVER (ORDER BY Total_Profit) / SUM(Total_Profit) OVER () AS Cumulative_Distribution
    FROM
        CategoryProfit
)
SELECT
    Product_Category,
    Total_Profit,
    Cumulative_Profit,
    Cumulative_Distribution
FROM
    CumulativeDistribution
ORDER BY
    Cumulative_Distribution;
```

product_category character varying (50)	total_profit numeric	cumulative_profit numeric	cumulative_distribution numeric
Sports & Outdoors	505718.00	505718.00	0.12598762988508553376
Games	673993.00	1179711.00	0.29389697981753495054
Art & Crafts	753354.00	1933065.00	0.48157723823121357618
Electronics	1001437.00	2934502.00	0.73106148460810821247
Toys	1079527.00	4014029.00	1.00000000000000000000

ANALYZE THE EFFICIENCY OF INVENTORY TURNOVER FOR EACH STORE BY CALCULATING THE INVENTORY TURNOVER RATIO

```sql
WITH Sales_COGS AS (
    SELECT
        s.Store_ID,
        SUM(s.Units * CAST(REGEXP_REPLACE(p.Product_Cost, '[^\d\.]', '', 'g') AS NUMERIC)) AS Total_COGS
    FROM
        Sales s
    JOIN
        Products p ON s.Product_ID = p.Product_ID
    WHERE
        EXTRACT(YEAR FROM s.Date) = 2023
    GROUP BY
        s.Store_ID
),

Avg_Inventory AS (
    SELECT
        Store_ID,
        ROUND(AVG(Stock_On_Hand), 2) AS Avg_Inventory
    FROM
        Inventory
    GROUP BY
        Store_ID
)
```

```sql
SELECT
    s.Store_ID,
    s.Total_COGS,
    a.Avg_Inventory,
    ROUND(
        CASE
            WHEN a.Avg_Inventory > 0 THEN s.Total_COGS / a.Avg_Inventory
            ELSE NULL
        END, 2
    ) AS Inventory_Turnover_Ratio
FROM
    Sales_COGS s
JOIN
    Avg_Inventory a ON s.Store_ID = a.Store_ID
ORDER BY
    Inventory_Turnover_Ratio DESC;
```

store_id integer	total_cogs numeric	avg_inventory numeric	inventory_turnover_ratio numeric
9	154098.20	16.54	9316.70
4	115003.48	12.49	9207.64
14	112988.13	12.66	8924.81
7	141100.86	15.94	8852.00
30	162213.79	20.83	7787.51
39	131164.10	17.31	7577.36
17	157713.91	20.90	7546.12
33	116104.14	15.40	7539.23
19	100813.95	14.17	7114.60
37	127996.26	18.17	7044.37
28	98485.11	14.48	6801.46
29	98543.89	15.00	6569.59
1	98757.66	16.09	6137.83
25	114356.90	18.83	6073.12
31	188110.15	31.55	5962.29
22	99192.42	16.80	5904.31
13	108267.42	18.72	5783.52
41	119758.96	20.71	5782.66
42	124723.45	21.60	5774.23
8	90192.29	15.66	5759.41
16	95744.56	16.63	5757.34
10	107818.01	18.77	5744.17
50	101790.84	18.31	5559.30
27	91646.20	16.52	5547.59


Thank You